



# *The road of a bug hunter*

*Moti Joseph*



# Who Am I?



- Sr Security researcher @ COSIENC
- (Previously worked at Websense, Checkpoint)

Hunting for vulnerabilities

Reverse engineering Microsoft patches

Writing plug-in for IDA and OllyDbg



# Agenda



In the next 45 minutes, we will cover:

- Introduce past zero-day exploits
- Discuss how software vulnerabilities are found
- Why attackers hunt for zero-days
- What do they need to look for before finding vulnerabilities in the Windows OS?
- where do they need to look at?
- how much time do they need to spend and which skills and knowledge do they need to know.
- We will answer this from a hacker and a bug hunted perspective.



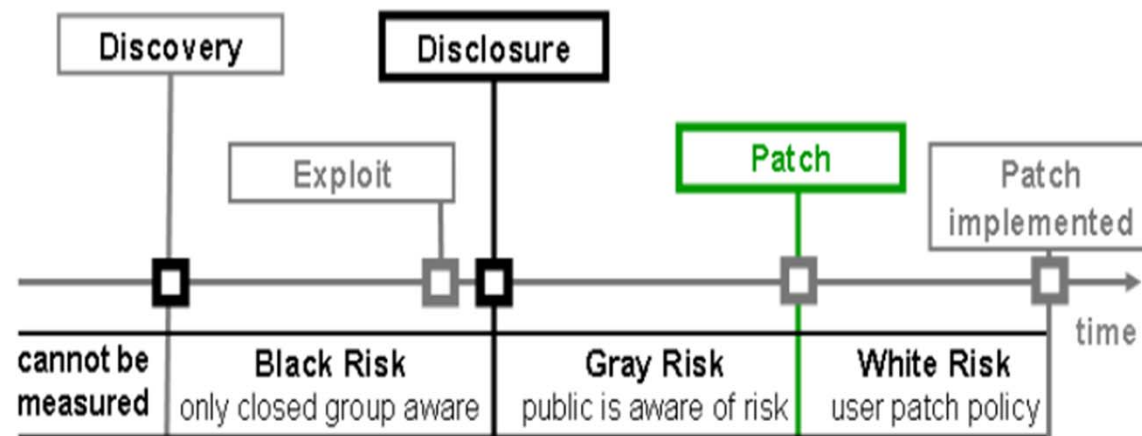


**A zero-day (or zero-hour) attack or threat is a computer threat that tries to exploit computer software vulnerabilities which are unknown to others, undisclosed to the software vendor, or without an available security fix**





## Lifecycle of a vulnerability





# Oday on the Wild



- [Windows URI Protocol Handling](#)  
Date Disclosed: 7/25/2007
- [MSN Messenger Video Conversation Heap Overflow](#)  
Date Disclosed: 1/31/2007
- [Microsoft DNS RPC Buffer Overflow](#)  
Date Disclosed: 4/7/2007
- [Windows .ANI Processing](#)  
Date Disclosed: 3/28/2007
- [Word Unspecified Exploit\(3\)](#)  
Date Disclosed: 1/25/2007





## More 0day

- [Microsoft Internet Explorer XML Processing](#)  
Date Disclosed: 11/15/2008
- [Microsoft Word XP/2002 SP3 Exploit](#)  
Date Disclosed: 7/8/2008
- [Microsoft Access Snapshot Viewer ActiveX](#)  
Date Disclosed: 7/7/2008
- [Microsoft Vulnerability in Server service](#)  
Date Disclosed: 10/15/2008
- [Excel Invalid Object](#)  
Date Disclosed: 2/24/2009





## Who hunt Oday

- Security Company ,(eEye,NGIS,ISS,NSFCUS.Secunia)
- Independed Researcher/Hackers  
***grey,black,white hat***
- Vendor



## *Who use 0day*



- Intelligence department
- Hackers
- Pen-tester
- Worms exploits



# Who is the target



- Military
- Business
- You ?Me? And everybody



# Why 0-day?



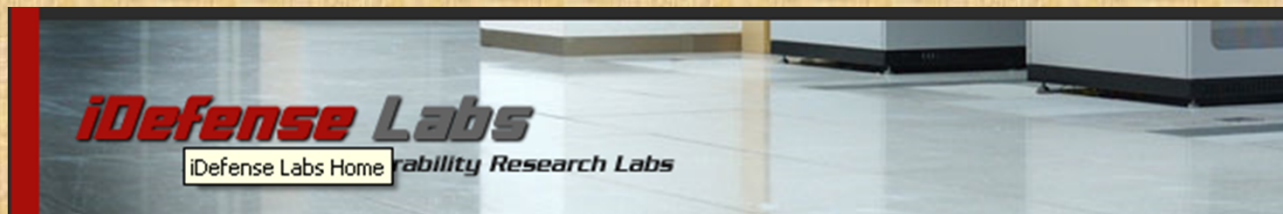
```
$ ls -l /usr/bin/efstool
-rwxr-xr-x 1 root root 14056 Sep 25 01:28 /usr/bin/efstool
$ /usr/bin/efstool perl -e 'print "A"x3000;'
Segmentation fault
$ gdb -q /usr/bin/efstool
(no debugging symbols found)...(gdb) run "perl -e 'print "A"x3000;'"
Starting program: /usr/bin/efstool perl -e 'print "A"x3000;'"
(no debugging symbols found)...(no debugging symbols found)...
(no debugging symbols found)...(no debugging symbols found)...
Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
(gdb) print $PC
$PC = 0x41414141
(gdb) x/48x ($PC-0x00)
0xbffff93: 0xbffff93 0xbffff7d0 0xbffffb46 0x4002463f
0xbfffd70: 0x00000003 0xbffff93 0xbffff7d0 0x00000000
0xbfffd80: 0xbfffd80 0x00000000 0x00000000 0x00000000
0xbfffd90: 0x00000000 0x00000000 0x00000000 0xbfffd70
0xbffdda0: 0x00000000 0x00000000 0x00000000 0x00000000
0xbffddb0: 0x00000000 0x00000000 0x00000000 0x00000000
0xbffddc0: 0x00000000 0xbffdd0 0x00000000 0x00000000
0xbffddd0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffdde0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffddf0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffde00: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffde10: 0x41414141 0x41414141 0x41414141 0x41414141
(gdb) quit
The program is running. Exit anyway? (y or n) y
```

# HACKING

## THE ART OF EXPLOITATION



They will buy it?





*But some will never sell !*





# How to hunt 0day



- Source code audit
- Binary Audit
- Fuzzing



# Hunting 0Days



- Reading Assembly
- Knowledge of the OS API
- Tools
- Experience
- Right place @ right time



# One day in the cracking world

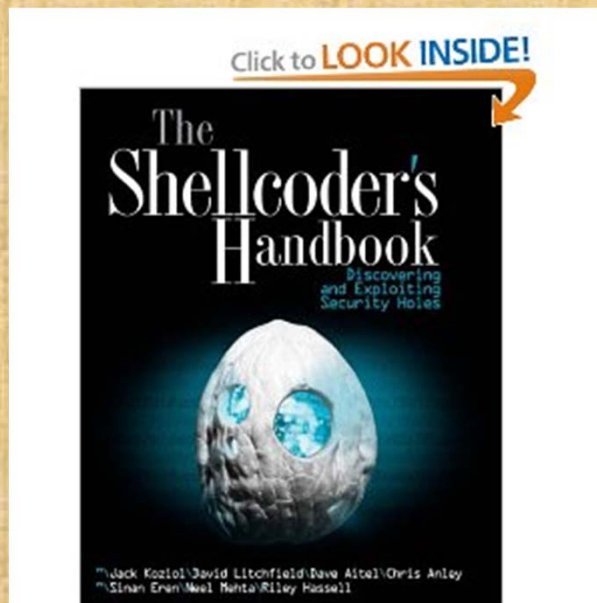
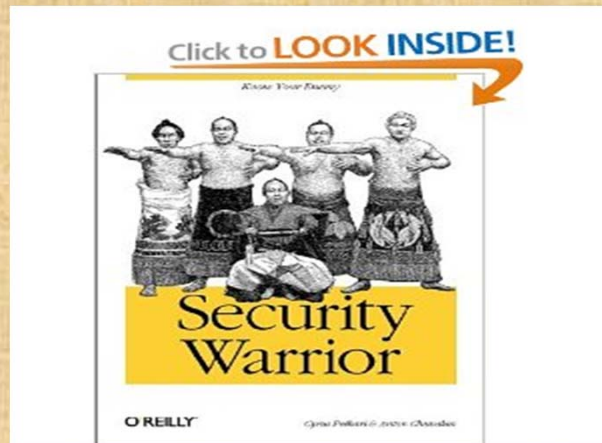


## Reading Assembly





# Knowledge of the OS API





# Tools





# Read Technical Advisories



## Security Advisories

The eEye Digital Security Research Team is dedicated to discovering new vulnerabilities and responsibly reporting them to the vendor via CERT coordination. This advisory information is meant to serve as a "time capsule" of the eEye Research vulnerabilities from the past and is intended solely as technical, in-depth analysis of the various vulnerabilities discovered by the eEye Research Team. This advisory information provided here adheres to eEye's responsible disclosure policy and supports the Company's goal to eliminate security vulnerabilities within computing networks.

**7/10/2009**

[eEye Retina Wireless Scanner .RWS File Processing Memory Corruption](#)

**11/20/2007**

[BitDefender Online Scanner 8 Double Decode Heap Overflow](#)

**11/15/2007**

[Multiple Vulnerabilities In .FLAC File Format and Various Media Applications](#)

**10/11/2007**

[CA BrightStor ARCserve Backup Server Arbitrary Pointer Dereference](#)

**9/20/2007**

[Multiple Vulnerabilities in CA ARCserve for Laptops and Desktops](#)

**8/14/2007**

[Windows Metafile AttemptWrite Heap Overflow](#)

**8/14/2007**

[VGX.DLL Compressed Content Heap Overflow Vulnerability](#)

**7/10/2007**

[Microsoft Publisher 2007 Arbitrary Pointer Dereference](#)

**7/5/2007**

[Sun Java WebStart JNLP Stack Buffer Overflow Vulnerability](#)

**6/8/2007**

[Yahoo! Webcam ActiveX Controls Multiple Buffer Overflows](#)



# Example Technical Advisory



## Technical Details:

URLMON.DLL version 6.0.2800.1565, distributed with the MS06-042 patch for Internet Explorer 6 SP1 on Windows 2000 and Windows XP SP1, contains a heap buffer overflow vulnerability due to an incongruous use of `lstrcpynA`. `CMimeFt::Create` allocates a 390h-byte heap block for a new instance of the `CMimeFt` class, within which there is a 104h (`MAX_PATH`)-byte ASCII string buffer at offset +160h:

```
1A4268DD push 390h ; cb
1A4268E2 call ??2@YAPAXI@Z ; operator new(uint)
```

When an access to a URL elicits a GZIP- or deflate-encoded response from the web server, `CMimeFt::Start` will attempt to copy the URL into the 104h-byte string buffer using the `lstrcpynA` API function, but it passes a maximum length argument of 824h (2084 decimal), a value typically used as the maximum length of a URL:

```
1A426199 push 824h ; iMaxLength
1A42619E push eax ; lpString2
1A42619F add esi, 160h
1A4261A5 push esi ; lpString1
1A4261A6 call ds:lstrcpynA
```

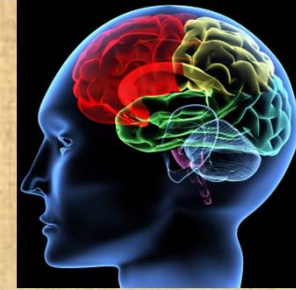
As a result, fields within the `CMimeFt` class instance as well as the contents of adjacent heap blocks can be overwritten with attacker-supplied data from the malicious URL.

URLMON.DLL in the MS06-042 patch for Internet Explorer 5 uses `MAX_PATH` both as the buffer size and as the maximum copy length, while URLMON.DLL in the patch for Windows XP SP2 and Windows 2003 uses 824h in both places.

This issue was originally documented as an Internet Explorer crash in Microsoft Knowledge Base Article KB923762 (<http://support.microsoft.com/?kbid=923762>; Revision 2.0 as of August 21st), in response to numerous reports of conflicts between the MS06-042 patch and various HTTP-based software products, dating back to at least August 11th. eEye independently discovered the flaw on August 15th and subsequently reported it to Microsoft on the 17th.



# The mind



- why me?
- is there a *vulnerability* there at all?
- someone already found something here
- this look too hard
- it will take me too much time
- Maybe I should try something easier

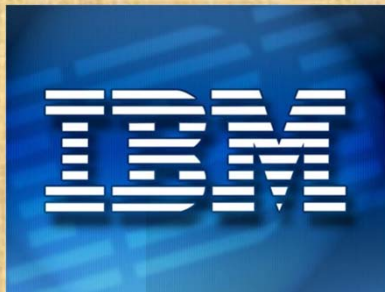
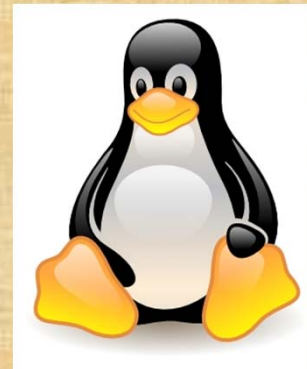


# Hunting for how long ?



Photoshop PSD file download - Resolution 1280x1024 px - [www.psdgraphics.com](http://www.psdgraphics.com)









# Upcoming advisories?

## Upcoming Advisories

The following is a list of vulnerabilities discovered by TippingPoint Zero Day Initiative researchers that are yet to be publicly disclosed. The affected vendor has been contacted on the specified date and while they work on a patch for these vulnerabilities, TippingPoint customers are protected from exploitation by IPS filters delivered ahead of public disclosure. TippingPoint customers are additionally protected against 0day vulnerabilities discovered by our own [DVLabs](#) researchers. A list of published advisories discovered by TippingPoint's DVLabs research group is available from:

<http://dvlabs.tippingpoint.com/advisories/upcoming/>

There are currently **90** advisories pending public disclosure.

| ZDI ID                               | Affected Vendor(s)        | Severity | Reported                 | Deadline   |
|--------------------------------------|---------------------------|----------|--------------------------|------------|
| ZDI-CAN-1020                         | <a href="#">Microsoft</a> | CVSS: 9  | 2011-02-28 (21 days ago) | 2011-08-27 |
| <i>Discovered by: Damian Put</i>     |                           |          |                          |            |
| ZDI-CAN-1112                         | <a href="#">Symantec</a>  | CVSS: 10 | 2011-02-17 (32 days ago) | 2011-08-16 |
| <i>Discovered by: Luigi Auriemma</i> |                           |          |                          |            |
| ZDI-CAN-1111                         | <a href="#">Symantec</a>  | CVSS: 10 | 2011-02-17 (32 days ago) | 2011-08-16 |
| <i>Discovered by: Luigi Auriemma</i> |                           |          |                          |            |
| ZDI-CAN-1110                         | <a href="#">Symantec</a>  | CVSS: 10 | 2011-02-17 (32 days ago) | 2011-08-16 |
| <i>Discovered by: Luigi Auriemma</i> |                           |          |                          |            |



# Let's go hunting

*Binary Audit*





# Types of Security Vulnerabilities



- ❖ ***Buffer Overflows***
- ❖ ***Heap Overflows***
- ❖ ***Insecure File Operations***
- ❖ ***Unvalidated Input***
- ❖ ***Race Conditions***
- ❖ **Memory Corruption**



# How do you look for 0day?



- ❖ *Find a vuln code and then trigger the code*
- ❖ **Trigger a code section and then check for vuln in the code**





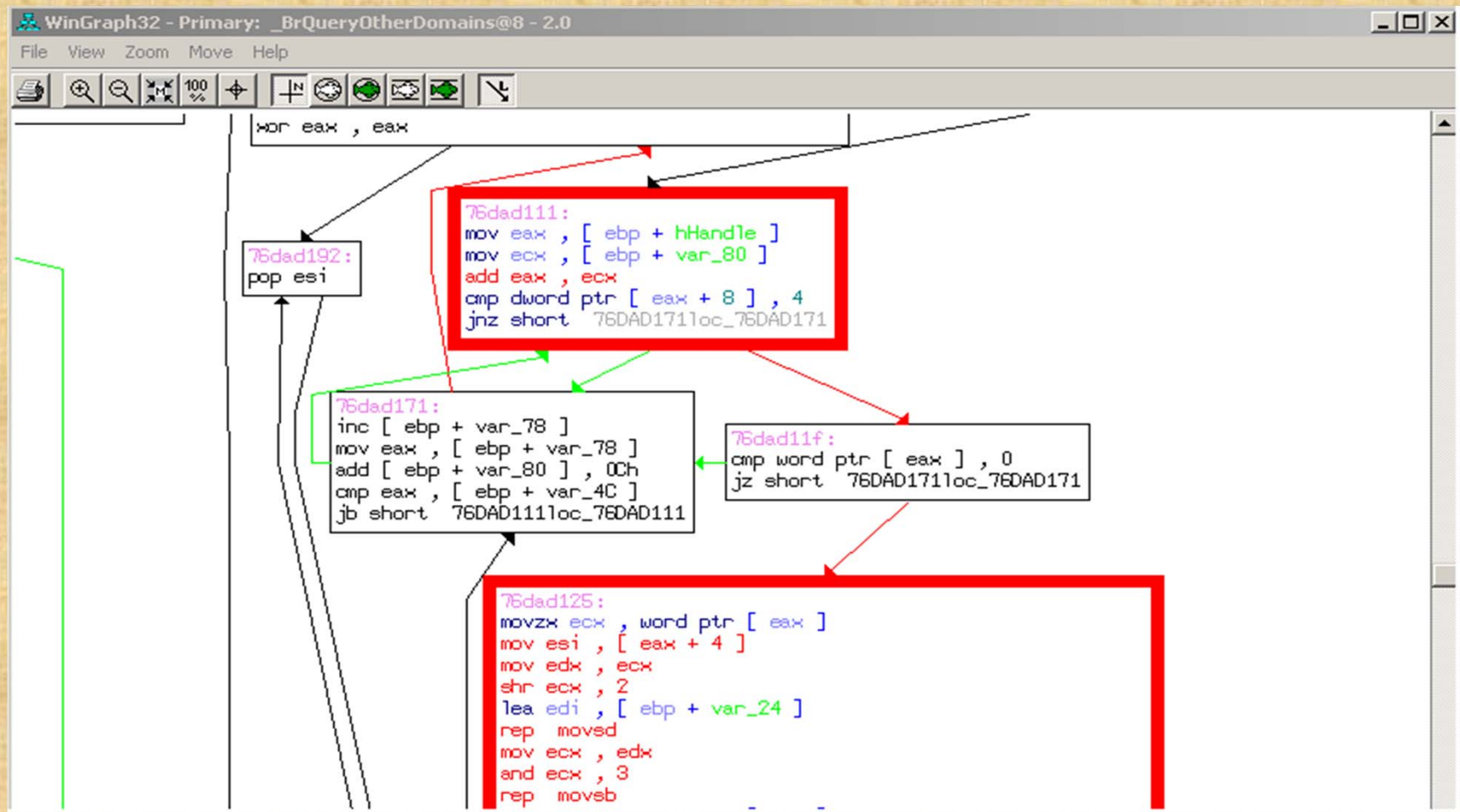
## #1 inline strcpy

```
text:66E97DC8 loc_66E97DC8: ; CODE XREF: CWMATCHBitmap(x,x,x,x,x)+4C63↓j
text:66E97DC8      mov     eax, [ebp+arg_4]
text:66E97DCB      mov     eax, [eax+edi*4]
text:66E97DCE      cmp     eax, [ebp-8]
text:66E97DD1      ja      short loc_66E97DF9
text:66E97DD3      test    eax, eax
text:66E97DD5      jz      short loc_66E97DF9
text:66E97DD7      dec     eax
text:66E97DD8      imul    eax, [ebp+arg_0]
text:66E97DDC      lea     eax, [eax+esi+54h]
text:66E97DE0      mov     edx, ebx
text:66E97DE2      sub     edx, eax
text:66E97DE4
text:66E97DE4 loc_66E97DE4: ; CODE XREF: CWMATCHBitmap(x,x,x,x,x)+4C7A↓j
text:66E97DE4      mov     cl, [eax]
text:66E97DE6      mov     [edx+eax], cl
text:66E97DE9      inc     eax
text:66E97DEA      test    cl, cl
text:66E97DEC      jnz     short loc_66E97DE4
text:66E97DEE      inc     edi
text:66E97DEF      add     ebx, 20h
text:66E97DF2      cmp     edi, [ebp+arg_C]
text:66E97DF5      jb      short loc_66E97DC8
text:66E97DF7      jmp     short loc_66E97E09
text:66E97DF9 ; -----
text:66E97DF9
text:66E97DF9 loc_66E97DF9: ; CODE XREF: CWMATCHBitmap(x,x,x,x,x)+4C5F↑j
text:66E97DF9      ; CWMATCHBitmap(x,x,x,x,x)+4C63↑j
text:66E97DF9      mov     dword ptr [ebp-4], 7E6h
text:66E97E00      jmp     short loc_66E97E09
text:66E97E02 ; -----
```



## #2 memcpy buffer overflow

**No** Boundary Check for string length >16 bytes







### Example #3 Fully Patched XP

**No** Boundary Check for length >0x4C4B40 bytes

```
73b60283 :
sub eax, ecx
add eax, edx
mov [ebp + pMem], eax
mov eax, 200h
cmp [ebp + pMem], eax
jge short 73B60297loc_73B60297
```

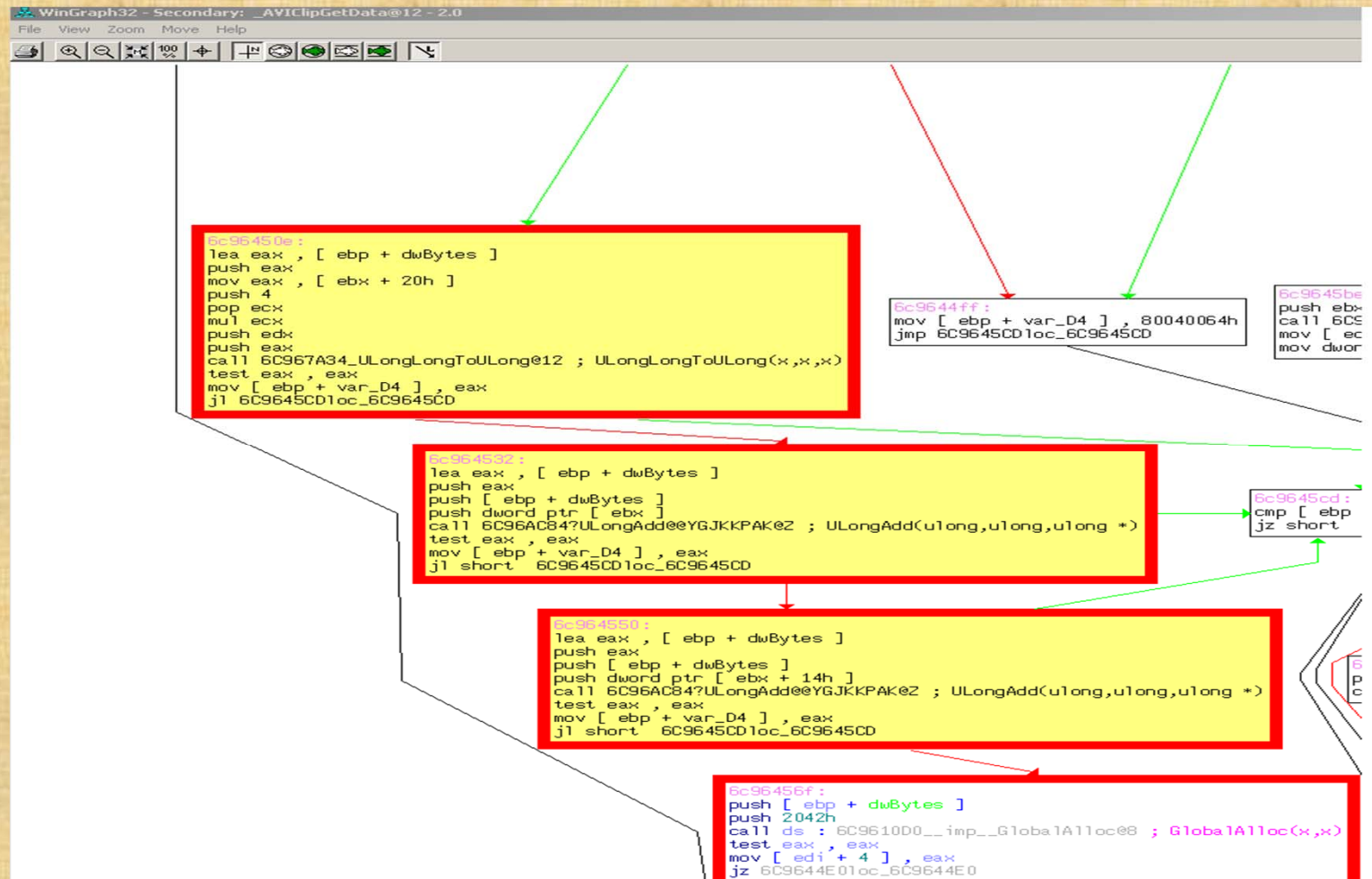
+ pMem], eax

```
73b60297 :
mov esi, ds : 73B51DAB__imp__GlobalHandle@4 ; GlobalHandle(x)
push edi ; pMem
call esi ; 73B51DABGlobalHandle(x) ; GlobalHandle(x)
push eax
call ds : 73B51DA4__imp__GlobalUnlock@4 ; GlobalUnlock(x)
mov eax, [edi + 4]
add eax, [ebp + pMem]
push 2002h
lea eax, [eax + eax * 2]
lea eax, ds : 8 [eax * 4]
push eax
push edi
call esi ; 73B51DABGlobalHandle(x) ; GlobalHandle(x)
push eax
call ds : 73B51DA0__imp__GlobalReAlloc@12 ; GlobalReAlloc(x,x,x)
push eax
call ds : 73B51DB0__imp__GlobalLock@4 ; GlobalLock(x)
test eax, eax
jz 73B603F4loc_73B603F4
```



## Example #4 Fully Patched Vista

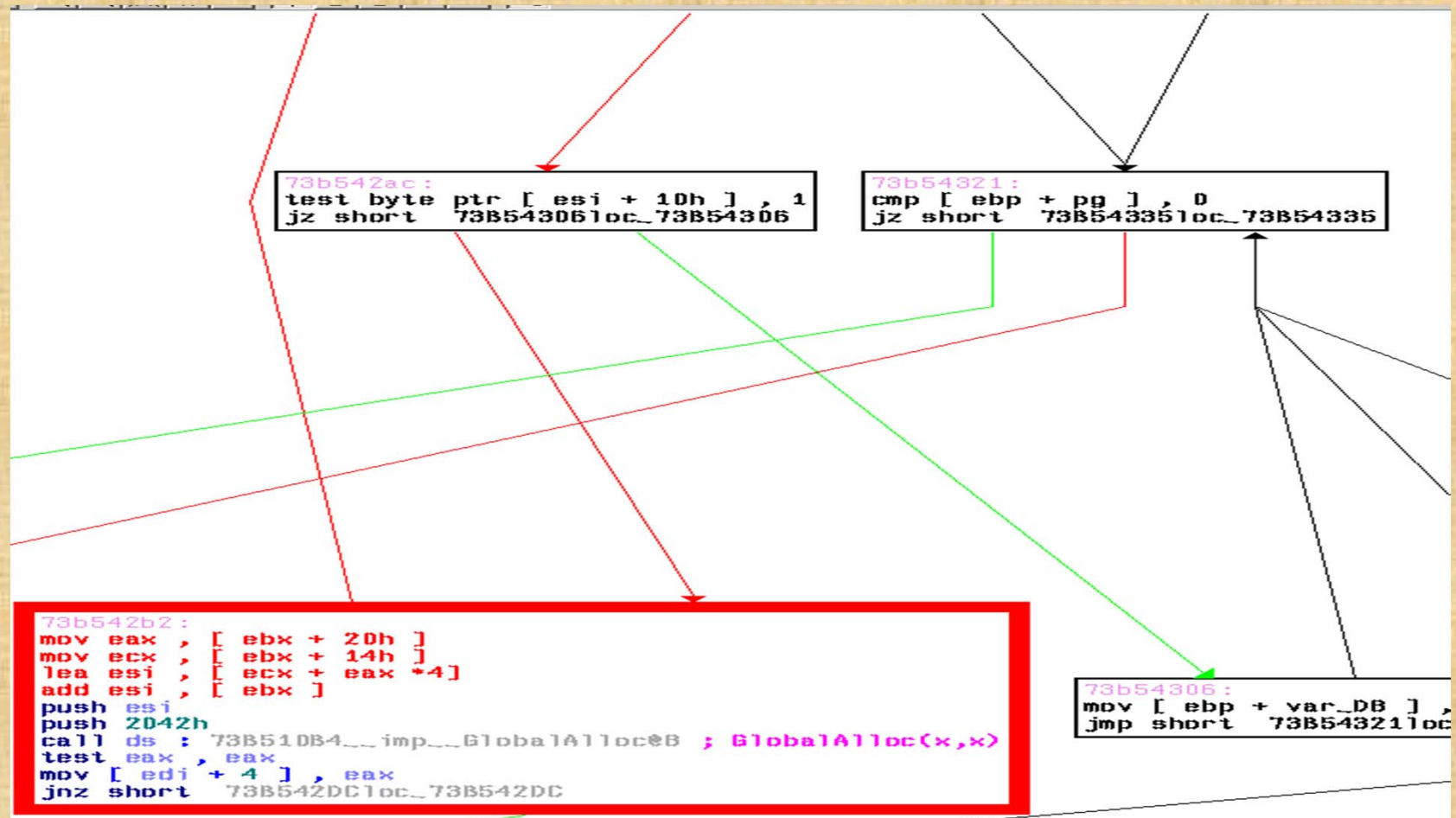
*Boundary Check* for INT overflow ULongAdd API





## Example #4 Fully Patched XP

NO ULongAdd API







## Example #5 Fully Patched Vista

### A Safe check for the DIB Size

```
ext:243CD620 jz     short loc_243CD60E
ext:243CD62C mov     eax, [edi+58h]
ext:243CD62F mov     [esi+58h], eax
ext:243CD632 mov     eax, [edi+5Ch]
ext:243CD635 mov     [esi+5Ch], eax
ext:243CD638 mov     eax, [edi+60h]
ext:243CD63B mov     [esi+60h], eax
ext:243CD63E
ext:243CD63E loc_243CD63E: ; CODE XREF: ConvertSurfaceDescTo
ext:243CD63E lea     eax, [ebp+arg_0]
ext:243CD641 push    eax
ext:243CD642 push    ebx
ext:243CD643 call    _SAFE_DIBSIZE@8 ; SAFE_DIBSIZE(x,x)
ext:243CD648 test    eax, eax
ext:243CD64A jl      short loc_243CD6A4
ext:243CD64C mov     eax, [ebp+arg_0]
ext:243CD64F mov     ecx, [ebp+var_8]
ext:243CD652 mov     [ebx+14h], eax
ext:243CD655 mov     [ecx+28h], eax
ext:243CD658 mov     eax, [ebp+arg_8]
ext:243CD65B test    eax, eax
ext:243CD65D mov     dword ptr [ecx+20h], 1
ext:243CD664 jz      short loc_243CD685
ext:243CD666 mov     ecx, [eax+8]
ext:243CD669 sub     ecx, [eax]
ext:243CD66B lea     edi, [esi+10h]
ext:243CD66E mov     [esi+8], ecx
ext:243CD671 mov     ecx, [eax+0Ch]
ext:243CD674 sub     ecx, [eax+4]
ext:243CD677 mov     [esi+0Ch], ecx
ext:243CD67A mov     ecx, [ebp+var_8]
ext:243CD67D mov     esi, eax
```



## Example #6 Fully Patched XP INT OVERFLOW



```
FF          mov     edi, edi
EC          push    ebp
            mov     ebp, esp
            push    ebx
5D 08       mov     ebx, [ebp+arg_0]
43 40       mov     eax, [ebx+40h]
C0 18       add     eax, 24
            push    eax                ; cb
15 50 12 6D 6E call    ds:__imp__CoTaskMemAlloc@4 ; CoTaskMemAlloc(x)
D0          mov     edx, eax
D2          test    edx, edx
55 08       mov     [ebp+arg_0], edx
07          jnz     short loc_6E6EB2DA
0E 00 07 80 mov     eax, 8007000Eh
62          jmp     short loc_6E6EB33C
            ; -----
            loc_6E6EB2DA:                ; CODE XREF: ConvertVideoInfoToVideoInfo2(x)
73 44       push    esi
            mov     esi, [ebx+44h]
0C          push    edi
            push    0Ch
            pop     ecx
FA          mov     edi, edx
A5          rep movsd
06          push    6
C0          xor     eax, eax
            pop     ecx
7A 30       lea     edi, [edx+30h]
AB          rep stosd
4B 40       mov     ecx, [ebx+40h]
```



## Example #7 Fully Patched XP



```
push esp
mov ebp, esp
push esi
mov esi, ecx
xor eax, eax
cmp [esi + 4], eax
jz short 6FD41BAAloc_6FD41BAA
```

```
6fd41baa:
cmp dword ptr [esi], 0FFFFFFFFh
jnz short 6FD41BA6loc_6FD41BA6
```

```
6fd41baf:
push eax
push [ebp + dwFlagsAndAttributes]
push 3
push eax
push 1
push 80000000h
push [ebp + lpFileName]
call ds:6FD4100C__imp__CreateFileW@28 ; CreateFileW(x,x,x,x,x,x,x)
xor ecx, ecx
cmp eax, 0FFFFFFFFh
setnz cl
mov [esi], eax
mov eax, ecx
```

```
6fd41ba6:
xor eax, eax
jmp short 6FD41BD2
```

```
6fd41bd2:
```



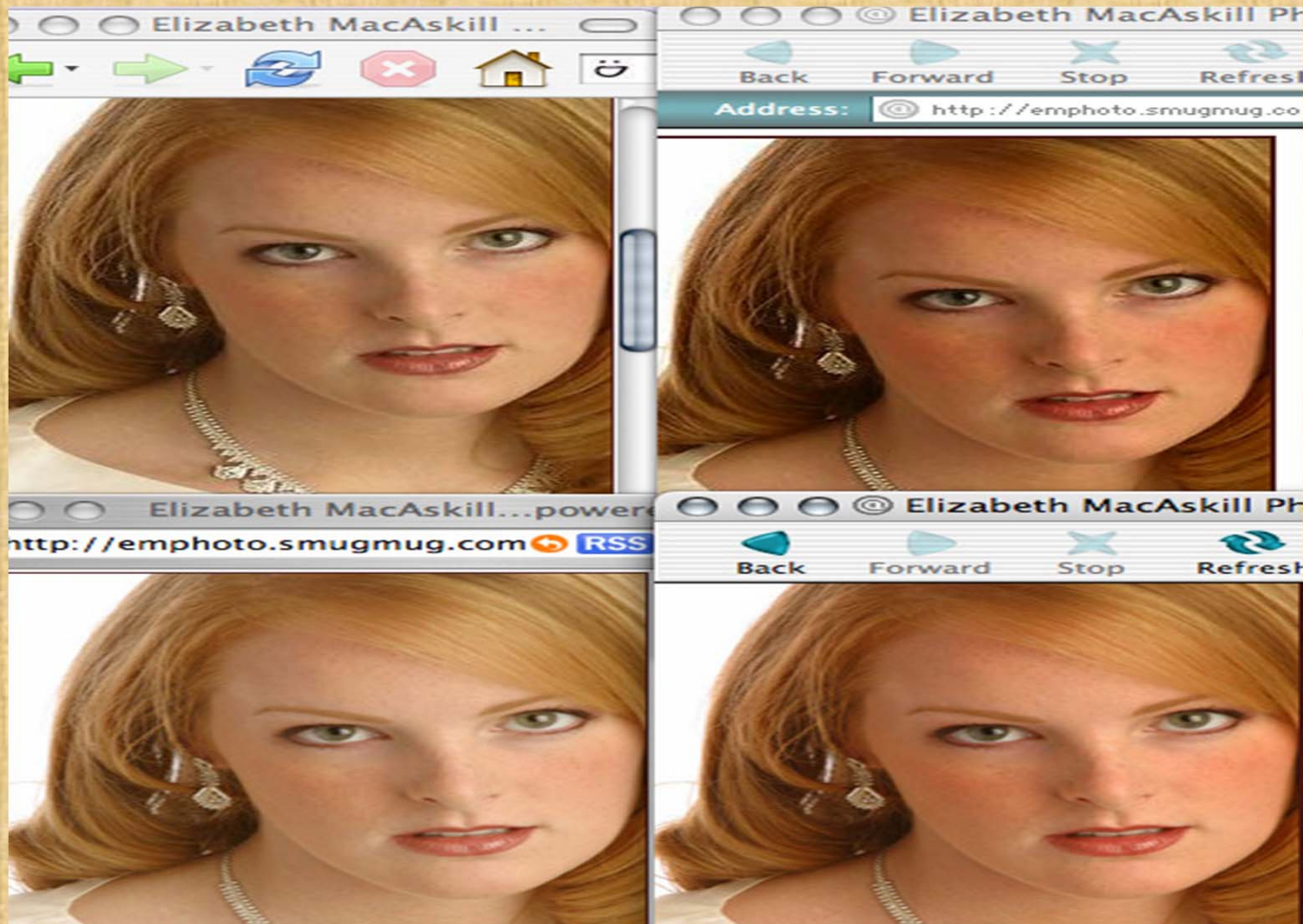
# **Microsoft Windows Color Management module buffer overflow**



**Image formats (such as TIFF, JPEG, PNG, EPS, PDF, and SVG) may contain embedded color profiles but are not required to do so by the image format. The International Color Consortium standard was created to bring various developers and manufacturers together. The ICC standard permits the exchange of output device characteristics and color spaces in the form of metadata. This allows the embedding of color profiles into images as well as storing them in a database or a profile directory.**



# Color profile embedded in PNG image





# Windows Color Management API



The DLL `icm32.dll` module is an element of the Windows Image Color Management (ICM) system. ICM is the Advance Programming Interface (API) used by applications to take advantage of color management capabilities in Windows. Another file associated with the `icm32.dll` application is the International Color Consortium (ICC) profile. The profile describes the color characteristics of a device. These files are used by the `icm32.dll` application to obtain reasonable color consistency across devices.



## Browser Test Results



- Firefox 3.5 Colour Management Enabled for all images
- IE6, IE7, IE8 , Chrome 0.\*, Firefox 3 with Colour Management Disabled, Opera 9.51
- **Internet Explorer 9 supports embedded ICC version 2 and version 4 color profiles in images**





## Internet Explorer 9: Windows XP users, you're out of luck

March 15, 2011 at 10:33 am by Todd Bishop 19

Microsoft last night released Internet Explorer 9, the latest version of its widely used web browser. Download it [here](#), but keep in mind that it only works with Windows Vista or Windows 7. The browser isn't compatible with Windows XP, and Microsoft doesn't make IE for Mac (or Linux).





**Thanks for ur time**

**Any Question?**

